**Lab Practical #13:**

To develop network using distance vector routing protocol and link state routing protocol.

**Practical Assignment #13:**

1. **C/Java Program: Distance Vector Routing Algorithm using Bellman Ford's Algorithm.**

import java.util.\*;

public class DistanceVectorRouting {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of nodes: ");

int n = sc.nextInt();

int[][] cost = new int[n][n];

int[][] dist = new int[n][n];

int[][] via = new int[n][n];

System.out.println("Enter cost matrix (999 for infinity):");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cost[i][j] = sc.nextInt();

dist[i][j] = cost[i][j];

via[i][j] = j;

}

}

// Bellman-Ford relaxation

for (int k = 0; k < n; k++) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (dist[i][j] > cost[i][k] + dist[k][j]) {

dist[i][j] = dist[i][k] + dist[k][j];

via[i][j] = k;

}

}

}

}

// Print final distance table

for (int i = 0; i < n; i++) {

System.out.println("\nRouting table for node " + (i + 1));

System.out.println("Destination\tNext Hop\tDistance");

for (int j = 0; j < n; j++) {

if (i != j) {

System.out.println((j + 1) + "\t\t" + (via[i][j] + 1) + "\t\t" + dist[i][j]);

}

}

}

}

}

**2. C/Java Program: Link state routing algorithm.**

import java.util.\*;

public class LinkStateRouting {

static final int INF = 999;

public static void dijkstra(int[][] cost, int n, int src) {

int[] dist = new int[n];

boolean[] visited = new boolean[n];

Arrays.fill(dist, INF);

dist[src] = 0;

for (int count = 0; count < n - 1; count++) {

int u = -1, min = INF;

for (int i = 0; i < n; i++) {

if (!visited[i] && dist[i] < min) {

min = dist[i];

u = i;

}

}

visited[u] = true;

for (int v = 0; v < n; v++) {

if (!visited[v] && cost[u][v] != INF && dist[u] + cost[u][v] < dist[v]) {

dist[v] = dist[u] + cost[u][v];

}

}

}

System.out.println("Shortest paths from node " + (src + 1));

for (int i = 0; i < n; i++) {

System.out.println("To " + (i + 1) + " -> " + dist[i]);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of nodes: ");

int n = sc.nextInt();

int[][] cost = new int[n][n];

System.out.println("Enter cost matrix (999 for infinity):");

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cost[i][j] = sc.nextInt();

}

}

for (int i = 0; i < n; i++) {

dijkstra(cost, n, i);

System.out.println();

}

}

}